Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Репин Дмитрий Юрьевич

Разработка системы локализации и распознавания дорожных знаков

Бакалаврская работа

Научный руководитель:
д. ф.-м. н., профессор Терехов А. Н.

Рецензент:
ст. преп. Смирнов М. Н.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Chair of Software Engeneering

Dmitrii Repin

Development of a system by traffic sign localisation and recognition

Bachelor's Thesis

Scientific supervisor:
professor Andrey Terekhov

Reviewer:
associate professor Alexander Privalov

Saint-Petersburg
2017

Оглавление

Введение

Последнее время активно развивается направление искусственного интеллекта для вождения машин. Одной из важнейших частей автомобильного автопилота является получение информации об окружении: об объектах, таких как другие автомобили на дороге, пешеходы, велосипедисты, препятствия вроде бетонных блоков или упавшего на дорогу дерева, о параметрах системы, скорости и направлении движения, качества сцепления с дорожным покрытием, качестве самого дорожного покрытия (несомненно, эти параметры взаимосвязаны, но из одного нельзя вывести другой в общем случае), погодные условия, которые влияют как на сам автомобиль с автопилотом, так и на окружающие его объекты, что необходимо учитывать для предсказания их поведения и подготовки собственной реакции. Также не стоит забывать и о том, что можно назвать мета параметрами: различные правила дорожного движения. Эти правила могут меняться в зависимости от места нахождения автомобиля, от окружающих его условий: времени суток, погоды и многого другого. Важным объектом для определения текущих правил являются дорожные знаки. Вообще, дорожные знаки - это стационарные объекты и часто можно встретить мнение, что распознавать их с помощью компьютерного зрения нет необходимости. Действительно, такое распознавание - это достаточно трудоемкая задача, которая, вдобавок ко всему, еще и не решается со 100% точностью в общем случае. При этом, критики данного подхода достаточно разумно заявляют, что автопилот все равно имеет доступ к системе навигации и, так как местоположение знаков меняется не часто, то можно просто запомнить где какой знак стоит. Либо же поместить на каждый дорожный знак радиометку, которая будет точно сообщать какой знак здесь стоит. Радиометка не требует продвинутого алгоритма распознавания: она будет точно сообщать нужную информацию, более того, радиосигнал принять зачастую проще чем видео: ему не помешают кусты или ветви деревьев да и просто другие автомобили. Чем же плохи эти варианты? Особая метка на знаках плоха сразу по нескольким параметрам:

для нее нужно гарантировать надежность источника, иначе кто угодно сможет управлять действиями автопилота, что вряд ли закончится хорошо. Также такую метку нужно установить, причем на все дорожные знаки хотя бы там, где предполагается использование автопилота. И за работой этих меток придется следить постоянно, ведь если автопилот управляет автомобилем на основании данных от таких меток, то, в случае отсутствия сигнала он не сможет определить: это метка сломалась или здесь знака действительно нет. Да, все эти вопросы можно решить. Но будет ли это надежнее системы компьютерного зрения? Однозначно можно сказать что это не будет дешевле. Что же насчет заранее созданной базы дорожных знаков? Эта идея весьма неплоха, но и у нее есть свои недостатки. Во-первых, эту базу должен кто-то создавать и обновлять. Если у автопилота не будет средств визуального распознавания знаков, то придется рассчитывать либо на добровольцев людей, либо на специально нанятых сотрудников. Первый способ крайне ненадежен как по срокам, так и по точности: действительно, полагаться на сообщение только одного водителя нельзя, нужно набирать некоторое их количество. При достаточной мотивации количество водителей это не проблема для крупных и оживленных трасс, но что делать с остальными? Нанимать же специальных сотрудников крайне дорого, да и скорость обновления будет крайне низкой. Можно было бы попытаться собирать информацию о знаках у официальных служб, но это также весьма непросто и скорость никто не гарантирует. Во-вторых, автомобиль - это средство повышенной опасности и полагаться только на одну систему все равно нельзя. Таким образом, даже если проблемы с базой и ее обновлением можно решить, то все равно нужна какая-то страховка на случай, если система геолокации вдруг даст сбой. Хотя это гораздо более актуально для, собственно, вождения, но, раз уж для него у автопилота будут системы визуального распознавания объектов, то естественно добавить туда и распознавание знаков. А такая система однозначно будет, даже если точность и надежность систем геолокации кардинально улучшаться: мы ведь не можем повесить датчик на каждого человека, который может оказаться на дороге

или рядом с ней, на каждое дерево возле дороги, на каждый камень на обочине. Подведем итог: аналоги визуальной системы распознавания знаков не могут заменить ее полностью, как по причине собственных проблем, как инженерных, так и экономических, так и по причине необходимости подстраховки на случай если что-то пойдет не так.

Система визуального распознавания дорожных знаков - это частный случай давно известной задачи распознавания образов, которая в свою очередь является одной из центральных задач технологии компьютерного зрения, родившейся еще в 60-х годах прошлого века. Естественно, что различных решений для этой задачи существует множество. На данный момент не существует однозначно выигрывающего решения этой задачи, более того, для различных частных случаев лучшими являются различные решения. Усугубляет проблему то, что в большинстве этих решений есть большое количество параметров, напрямую влияющих на основные параметры работы системы: на точность, скорость работы, скорость обучения (там где оно нужно) и объем занимаемой памяти. Таким образом было принято решение перед разработкой непосредственно системы распознавания, разработать прототип для изучения подходов. Такой прототип имеет значительно более мягкие требования к вычислительным мощностям, скорости работы и может занимать гораздо больше памяти, чем итоговая система, которая сильно ограничена параметрами автомобиля, в который она устанавливается. Благодаря этому появляется возможность относительно быстро разработать систему для изучения различных алгоритмов на языке высокого уровня.

1. Обзор

1.1. Обзор существующих подходов к распознаванию и локализации

Распознавание образов - очень активно развивающееся направление, на данный момент придумано множество алгоритмов, с помощью которых можно решать поставленную задачу. Вот несколько самых распространенных алгоритмов и подходов для локализации и распознавания образов на изображении:

- Метод Виолы-Джонса
- Гистограммы ориентированных градиентов
- Визуальные словари
- Сверточные нейронные сети

Метод Виолы-Джонса

Алгоритм для локализации объектов на изображении. Изначально создавался для поиска лиц, но может применяться и для других классов объектов. Локализация с помощью данного алгоритма происходит быстро и достаточно эффективно благодаря трем ключевым особенностям: использованию интегрального изображения, использования алгоритма обучения, основанного на алгоритме AdaBoost [5] и использовании каскада классификаторов.[12] Интегральное изображение в точке (x, y) содержит сумму всех пикселей выше и левее этой точки включительно:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

где ii(x, y) - интегральное изображение, а i(x, y) - оригинальное. [12] Признаки используемые в данном алгоритме, это прямоугольные черно-белые области, в которых из суммы пикселей белых областей вычитается сумма пикселей черных областей. Пример признаков: Данный

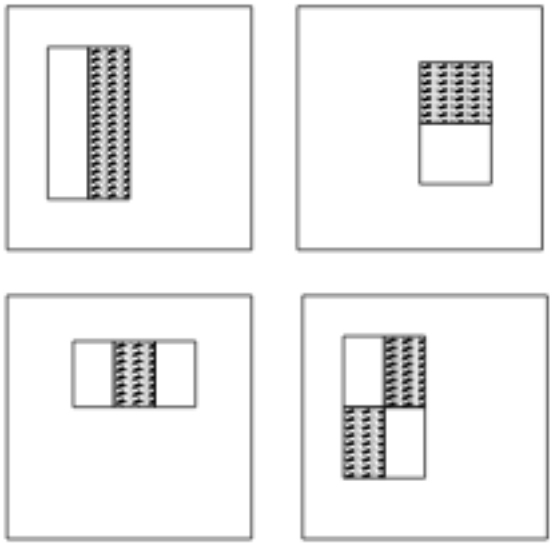


Рис. 1: Изображение взято из [12]

алгоритм работает очень быстро, но долго обучается и не для всех объектов возможно подобрать признаки, с хорошей точностью эти объекты локализующие на изображениях.

Гистограммы ориентированных градиентов

Метод основан на подсчете количества направлений градиента в локальных областях изображения. Основная идея в том, что внешний вид и форма объекта на участке изображения могут быть описаны распределением градиентов интенсивности или направлением краев даже без информации о точном местоположении данных градиентов или краев.[3] Преимущество данного алгоритма в поддержке инвариантности искажений объекта, кроме изменения его формы. Но при этом скорость работы данного метода ниже многих других. Изначально данный метод был разработан для поиска пешеходов, но может применяться и к другим классам объектов.

Визуальные словари

Метод основан на подходе, при котором особенности изображения трактуются как слова из из таких особенностей составляется словарь.

Такая интерпретация позволяет заменить объект на набор его свойств, и искать свойства, а не целый объект. Это дает возможность находить объект, даже если он был искажен, как по форме, так и по другим параметрам, например по размеру или освещенности. Еще одним преимуществом данного подходя является то, что он позволяет использовать практически без изменений подходы, используемые для текстового информационного поиска, такого, какой используется в поисковых машинах для интернета.[11] Главный недостаток такого подхода - скорость его работы.

Сверточные нейронные сети

Сверточная нейронная сеть - архитектура нейронной сети, которая базируется на чередовании сверточных слоев и субдискретизирующих слоев. Название архитектура сети получила из-за наличия операции свёртки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.[15] Эта архитектура была подробно описана в работе Яном Лекуном (Yann LeCun) в работе "Gradient-Based Learning Applied to Document Recognition"[6]. Преимущества такого подхода в высокой точности распознавания и локализации, высокая, хоть и не абсолютная, устойчивость к вариативности изображения, возможность значительно ускорить скорость как работы, так и обучения за счет распараллеливания вычислений. Среди недостатков самый значительный - это большое количество значимых параметров, хоть и не такого большого как для полносвязных нейросетей, без четкого алгоритма по определению лучшего сочетания. Зачастую многие параметры приходится определять эмпирически. Данный подход был выбран для прототипа системы из-за его успешности на соревнованиях "ImageNet Large Scale Visual Recognition Challenge"[7], которые являются, де-факто, главными в области распознавания образов.

2. Прототип -> продукт

Так как решений для задачи локализации образа и его распознавания придумано уже очень много, и разные решения могут имеют различную эффективность в различных ситуациях, то логично сначала провести исследовательскую работу и выбрать наилучший вариант, исходя из конкретных цифр. Для этого имеет смысл сначала разработать прототип, для которого не важны абсолютные значения скорости работы, но на котором можно сравнить различные параметры. Важные параметры для сравнения по степени важности для прототипа в порядке убывания:

1. Точность распознавания
2. Относительная скорость работы
3. Возможность дообучения
4. Скорость обучения

Самое важное для прототипа - точность распознавания. Это основной параметр и он не поменяется при реализации готовой системы. Точность зависит от выбранного алгоритма, его параметров и, для некоторых алгоритмов, в частности для выбранной мной системы, сильно зависит от набора изображений для обучения. Набор должен быть достаточно большой и разнообразный, чтобы нейросеть смогла выделить обобщающие признаки. Также важна скорость работы. Многие алгоритмы для распознавания реально довести до близкого уровня точности, но, если система, основанная на одном из алгоритмов, работает очень медленно, то ее применимость весьма ограничена. Скорость работы - это сложный параметр в случае с прототипом, так как итоговая система однозначно будет лучше оптимизирована. Поэтому абсолютная скорость работы - бессмысленный параметр, сравнивать можно только относительную скорость работы систем на разных алгоритмах. При прочих равных также имеет ценность возможность дообучения системы без разрабатывания новой с нуля. Ведь очевидно, что набор дорожных

знаков может изменяться и с этим надо будет как-то работать в будущем. Часть алгоритмов позволяет такое дообучение, часть - нет. Для исследовательской части работы можно выделить еще параметр скорости обучения системы. От него напрямую зависит продолжительность этого этапа работ. Выбирать подход основываясь на этом параметре неразумно - ведь для итоговой системы он не имеет никакого смысла, кроме случаев, конечно, когда обучение требует совершенно невозможных сроков. Но этот параметр позволяет расставить приоритеты для проверки различных моделей. Имеет смысл сначала проверить модели, обучение которых происходит быстро, если они потенциально подходят по выше упомянутым критериям.

3. Локализация и классификация

Проблему распознавания дорожных знаков можно разделить на две проблемы:

1. Проблема поиска дорожного знака на изображении или кадре видео (локализация)

2. Проблема определения конкретного знака (классификация)

С помощью сверточных нейросетей иногда возможно классифицировать объект на изображении даже без его локализации, но в случае с дорожными знаками это не получится. Во-первых, дорожных знаков на изображении может оказаться несколько, во-вторых, зачастую важно знать где конкретно находится каждый из этих знаков. Например, если поиск и классификация используется для автопилота, то необходимо отличать знаки на противоположной стороне дороги от своих и от знаков расположенных на других автомобилях. Таким образом я разрабатываю систему, которая решает эти две проблемы по отдельности.

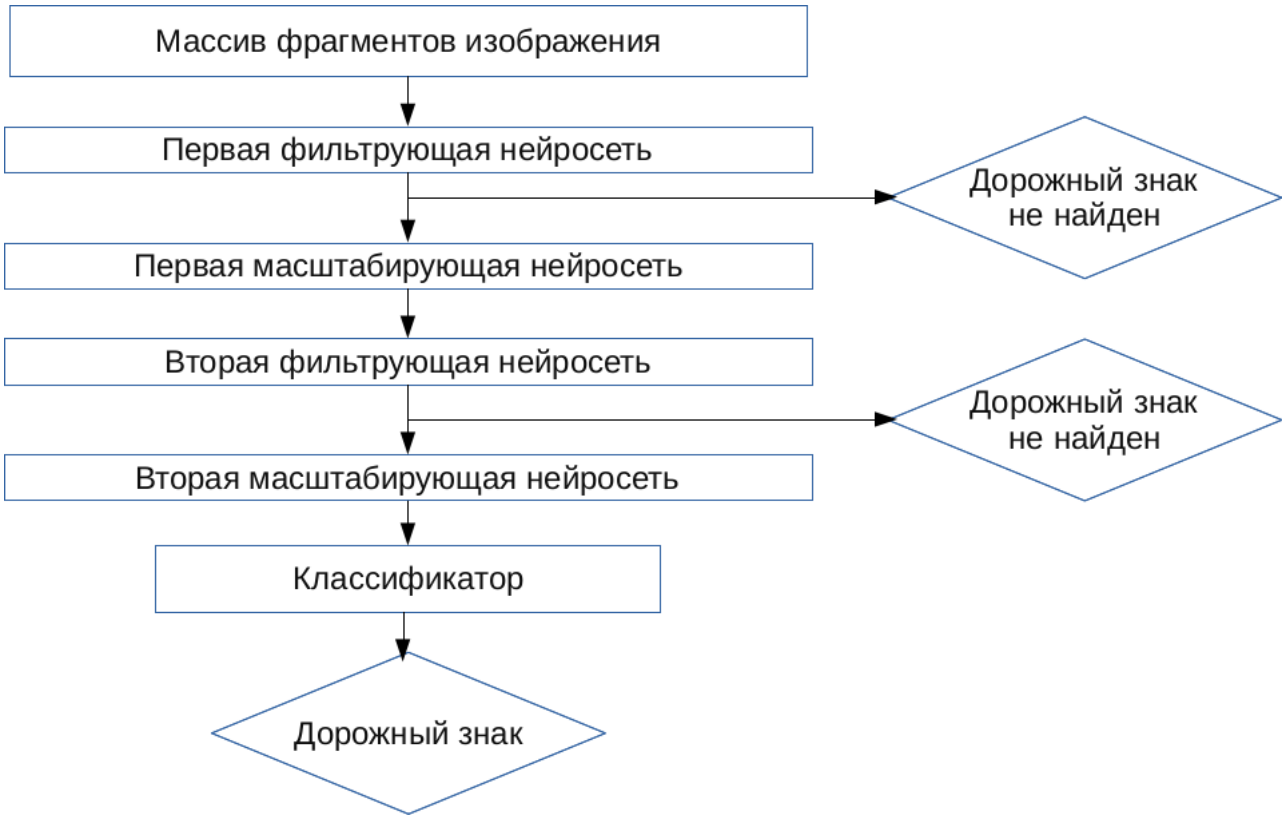


Рис. 2: Текущая архитектура

4. Общее описание системы

В текущем варианте реализации система выглядит как каскад нейросетей, подобный описанному в [1]. Данный подход позволяет достигать высокой точности, сохраняя при этом скорость работы. Также в данной архитектуре возможно разделить и выполнять параллельно обучение для различных частей системы, что уменьшает общее время обучения.

4.1. Общее описание

Система состоит из двух локализующих нейросетей, двух масштабирующих и одной квалифицирующей. Общая архитектура системы: данные поступают на вход первой локализующей сети. Эта сеть для каждого окна проверяет есть ли в нем знак или его часть. Часть для проверки знака задается на этапе обучения, это еще один параметр на-

стройки системы. На выходе получается набор окон, предположительно содержащий дорожный знак. Этот набор обрабатывается алгоритмом Non Maximum Supression и оставшиеся окна передаются первой масштабирующей сети. Первая масштабирующая сеть для каждого окна определяет класс изменения окна. Класс выбирается из заранее заданного набора множителей. После чего данные передаются во вторую локализационную сеть. Снова для каждого окна определяется есть ли в нем знак. Набор окон с положительным ответом еще раз прогоняется через Non Maximum Supression алгоритм и передается во вторую масштабирующую сеть. Принцип работы второй масштабирующей сети аналогичен первой. После второй масштабирующей сети фрагменты передаются в квалифицирующую сеть, которая уже определяет тип дорожного знака в этом окне.Такая архитектура имеет следующие преимущества: каскад нейросетей сам по себе позволяет разработать более мощную нейросеть на тех же вычислительных ресурсах, что и для одной но большой нейросети. Отдельные нейросети для масштабирования позволяют специализировать части системы. Эти сети отдельно обучаются и их можно отдельно настраивать не трогая остальное.

4.2. Локализационные сети

Локализационная цепочка состоит из двух сетей. Задача первой сети быстро отсеять много фрагментов изображения точно не содержащих дорожного знака. Фрагмент считается содержащим дорожный знак, если в него попадает не меньше определенного процента общей площади дорожного знака. В текущей реализации для упрощения расчетов считается площадь не конкретно знака, а минимального в пикселях прямоугольника, описывающего знак. Параметр процента площади настраивается и на данный момент составляет 80%. Так как на вход этой сети подается очень много фрагментов изображения, а содержащих дорожный знак среди них мало, то очевидны следующие требования:

- Сеть должна работать быстро

- Точность истинно отрицательных ответов важна, или, иначе говоря, важно отсутствие ложно отрицательных ответов
- Не сильно важна точность истинно положительных ответов, пока отсеивается достаточное число фрагментов изображения не так уж важно количество ложно положительных ответов среди оставшихся

Исходя из этих требований было решено подавать на вход сети разреженное изображение. Из исходного фрагмента берется только каждый четвертый пиксель в каждой четвертой строке. Это значительно уменьшает количество необходимых вычислений, но при этом содержит достаточное количество информации для распознавания, как показано в [1]. Конкретно в нашем случае первая локализационная сеть имеет размер 16x16 пикселей, созданное из исходного 64x64. В первом прототипе исследуется теория, согласно которой для распознавания знака важен цвет, следовательно в изображении будет три слоя RGB. Для второй локализационной сети задача похожая: условия по точности аналогичны, но, так как после отсеивания первой сетью фрагментов осталось значительно меньше, то можно увеличить количество вычислений. Вторая сеть таким образом обрабатывает также разреженные изображения, но уже не в 4 раза, а в два. Третья и последняя локализационная сеть - это финальная стадия локализации. Здесь уже скорость работы значительно менее актуальна чем точность. Это возможно благодаря тому, что большую часть изображений уже отсеяли предыдущие стадии. Эта сеть обрабатывает не разреженное изображение и у нее, потенциально, может быть больше слоев.

4.3. Масштабирующие сети

Проблема локализации не только в том, что необходимо найти объект на изображении, но еще и в том, что этот объект зачастую разные размеры на разных изображениях. Для решения этой проблемы необходимы дополнительные масштабирующие нейросети. В разрабатыва-

емой системе изначально таких нейросетей две. При проведении исследований возможно изменение их количества как в сторону уменьшения, так и в сторону увеличения. Так как в фрагментах, на которые разбито исходное изображение большинство, очевидно не содержат дорожных знаков, то применять масштабирующую сеть сразу к ним не имеет смысла - большая часть как была ненужным мусором, так им и останется. Следовательно имеет смысл ставить первую масштабирующую сеть только после первой локализационной, когда большая часть отрицательных изображений уже будет отсеяна. Здесь также все еще важна скорость обработки, поэтому первая масштабирующая сеть также как и вторая локализационная, работает с разреженным изображением. Вторая масштабирующая сеть находится перед третьей локализационной и, аналогично, мы можем позволить дать ей больше вычислительных ресурсов, поэтому она работает уже с полным изображением. Вторая сеть нужна для корректировки, первой. С одной стороны она в состоянии убрать часть ошибок первой, менее точной сети, с другой позволяет получить размеры изображения не попадающие в классы первой вообще, либо скорректировать ошибки первой менее точной нейросети.

4.3.1. Принцип работы масштабирующей нейронной сети

На вход подается изображение с параметрами (x, y, w, h), где (x, y) - координаты левого верхнего угла, а (w, h) - ширина и высота изображения. На выходе сеть возвращает один из заранее заданных вариантов преобразования окна. Варианты выглядят так:

$$(x - \frac{x_n w}{s_n}, y - \frac{y_n h}{s_n}, \frac{w_n}{s_n}, \frac{h_n}{s_n})$$

Варианты для x_n и y_n совпадают с вариантами приведенными в [1], но s_n отличается:

$$s_n \in \{0.5, 0.75, 1.0, 1.5, 2.0\}$$

$$x_n \in \{-0.17, 0, 0.17\}$$

$$y_n \in \{-0.17, 0, 0.17\}$$

4.4. Квалифицирующая сеть

Квалифицирующая сеть в данном случае - достаточно простая сверточная нейронная сеть с парой сверточных слоев. На выходе присваивает изображению класс, соответствующий найденному дорожному знаку. Теоретически возможно имеет смысл разбить эту сеть на несколько, в таком случае первая будет определять не сам знак, а только его тип, после чего изображение будет передаваться в нейронную сеть для конкретизации знака. Но в текущей реализации нейросеть одна.

5. Выбор и обработка набора данных для обучения и тестирования модели

Еще одной проблемой при разработке системы распознавания, основанной на машинном обучении, является поиск размеченных данных, подходящих для обучения. Есть два выхода: можно либо найти готовый набор, либо разметить самостоятельно. Для большей эффективности необходимо размечать самостоятельно, причем на изображениях снятых на устройство, с которым позже будет работать готовая система. Это даст максимальную точность, ведь в таком случае систему можно обучить корректировать даже недостатки съемки. Но у такого подхода есть несколько недостатков: во-первых, набор данных должен быть большим, в нем должны встречаться все необходимые дорожные знаки и, желательно, в максимально широком диапазоне условий, таких как освещение, погодные условия, частичное перекрытие, искажение самого дорожного знака. Во-вторых, даже если такой набор данных получится собрать, то необходимо его каким-то образом разметить, а это долгий монотонный труд. Для этого необходима отдельная команда. Можно попытаться автоматизировать разметку, но для этого придется разработать систему практически идентичную необходимой, разве что без требований к скорости работы. Но можно взять готовый набор данных, для которого уже проведена работа по разметке. Такие наборы можно найти даже в свободном доступе, хотя их количество, очевидно, ограничено. Естественно что в уже готовом наборе будут недостатки: там может не оказаться необходимых знаков или количество изображений их содержащих будет слишком мало для адекватного обобщения нейросетью, устройство, на которое сняты изображения из набора может обладать техническими характеристиками, сильно отличающимися от целевого устройства. С другой стороны, для прототипа эти недостатки не критичны, так как нейросети обладают мощными обобщающими способностями, то их возможно будет обучить для готовой системы уже на подходящем наборе и получить ту же точность, что и у прототипа. Также недостаток данных можно скорректировать с помо-

щью добавления синтетических данных, что дает неплохие результаты, как показано в работе [4]. Для обучения своего прототипа я взял набор данных "LISA Traffic Sign Dataset" из [8]. Вместе с этим набором предоставляются инструменты для некоторой его настройки, с помощью этих инструментов возможно выбрать отдельно изображения сделанные на определенное устройство и/или изображения содержащие только определенные дорожные знаки. Также этот набор данных имеет удобную разметку с достаточным количеством данных для моей задачи. Перед началом работы было проведено небольшое исследование по размеру дорожных знаков, которые встречаются в наборе. Это необходимо для масштабирующей цепочки нейросетей: в этой цепочке мне необходимо знать возможные размеры для корректного определения набора множителей масштабирования и частоту встречающихся размеров для выбора первоначального размера фрагментов изображения. В выбранном наборе данных часть дорожных знаков встречается очень небольшое количество раз, чтобы гарантировать достаточное количество информации для обучения я выбрал несколько наиболее часто встречающихся знаков и работал только с изображениями содержащими эти знаки. Вообще для обучения нейросети необходимы в том числе и отрицательные примеры, то есть изображения не содержащие искомого объекта, но в данном случае единицей обучения является не все изображение целиком, а один его фрагмент, а фрагментов не содержащих знак даже на изображении где он есть - большинство, поэтому необходимости в дополнительных изображениях нет.

6. Тестирование полученного результата

Система тестирования модели - критически важный компонент разрабатываемой системы. Без этого компонента невозможно оценить насколько разрабатываемая система эффективна и выполняет ли она вообще свои задачи. Тестирование модели происходит на данных, не использованных в обучении, в качестве метрики для оценки качества обучения модели была выбрана f1 метрика.

F1 метрика

Для определения качества обучения локализационной нейросети недостаточно оценивать результат только по характеристике верно или не верно был дан ответ. Всего возможно 4 варианта ответа:

1. Истинно положительный (tp) - когда объект есть на фрагменте и нейросеть его там нашла
2. Истинно отрицательный (tn) - объекта нет и нейросеть отвергла фрагмент
3. Ложно положительный (fp) - объекта на фрагменте нет, но нейросеть говорит что есть
4. Ложно отрицательный (fn) - объект есть, но нейросеть его не нашла

Ложно положительные ответы (fp) называют ошибкой первого типа, ложно отрицательные (fn) - ошибкой второго типа. Обозначим

$$precision = \frac{tp}{tp + fp} \tag{1}$$

$$recall = \frac{tp}{tp + fn} \tag{2}$$

Где Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов

положительного класса из всех объектов положительного класса нашел алгоритм. F-мера (в общем случае F_β) — среднее гармоническое precision и recall:

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \tag{3}$$

β в данном случае определяет вес точности в метрике, и при $\beta = 1$ это среднее гармоническое (с множителем 2, чтобы в случае precision = 1 и recall = 1 иметь $F_1 = 1$) [14]

7. Режим разработки

Для успешной разработки продукта полезно использовать один из фреймворков разработки. Таких фреймворков много и, естественно, даже выбрав конкретный, не стоит слепо следовать ему во всем. Но все же выбор парадигмы разработки может сильно помочь проекту определяя важные ключевые точки для контроля состояния разработки, помогая оценить ее продолжительность и описывая методы с помощью которых разработка ведется. Рассмотрим наиболее распространенные из них:

- Чистая комната
- Водопадная модель
- V модель
- Итерационная модель
- Спиральная модель
- Прототипирование
- Адаптивная разработка(Agile)

Чистая комната

Парадигма предназначенная для разработки программного обеспечения с определенным уровнем надежности. Опирается на формальные методы, основанные на математическом аппарате, для всех шагов разработки. Главной задачей данного фреймворка является предотвращение ошибок, вместо их исправления, что достигается в первую очередь ручной математической верификацией.[10] Эта задача, как и метод ее достижения, очевидным образом не подходит для разработки прототипа системы машинного зрения.

Водопадная модель

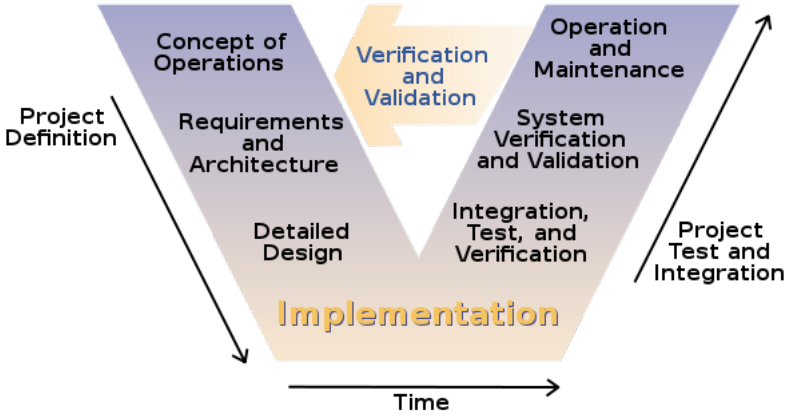
Методология, предполагающая последовательный не итеративный процесс. Впервые была описана в работе "Managing the Development of Large Software Systems" [13] Состоит из нескольких шагов:

1. Анализ требований и составление формальной спецификации
2. Разработка архитектуры и дизайна продукта
3. Разработка продукта
4. Тестирование
5. Выпуск продукта
6. Поддержка продукта

При этом переход к следующему шагу происходит только при полном выполнении предыдущего. С этим подходом легко оценивать сроки разработки и ее прогресс благодаря четко прописанным ключевым точкам. Повышается качество продукта из-за больших этапов сбора и анализа требований и дизайна продукта. Ошибки выловленные на этих этапах гораздо дешевле исправлять, чем пойманное на этапе тестирования. Но все преимущества требуют неизменности требований, так как при их изменении ломается вся цепочка разработки и приходится возвращаться к самому началу. Для данного проекта для стадии прототипирования и исследования такая модель в принципе не подходит. Данная стадия может являться первым или вторым этапом разработки по водопадной модели, но сама не может ее использовать.

V модель

Расширение водопадной модели. В отличие от классической водопадной модели процесс идет не строго вниз от общих абстракций к частностям, но начинает подниматься обратно после фазы кодирования. Общий вид процесса представлен на изображении:



Данная модель имеет ряд преимуществ:

- Четко определяются стадии процесса и их логические связи
- Уделено большое внимание разработке тестовой документации в наиболее ранних стадиях, на которых это возможно
- Тестированию уделяется столько же внимания, сколько и непосредственной разработке
- Описывается простой и понятный план разработки продукта

К сожалению, недостатки данной модели настолько перевешивают ее преимущества, что в настоящее время она практически не используется. Недостатки V модели:

- Данная модель не предоставляет механизмов для работы с изменениями в проекте. При этом абсолютное большинство реальных проектов изменяются за время разработки
- Необоснованная изоляция различных уровней тестирования, например таких как юнит тестирование и интеграционное тестирование.
- Чрезмерное упрощение процесса разработки, что может привести к ложному чувству уверенности в контроле за ситуацией

Модель неплоха как академический концепт, но не выдержала испытание практикой. Использовать эту модель для проекта было бы неразумно.

Итерационная модель

Эта модель берет за основу водопадную методологию, но вместо одного большого релиза происходит несколько более мелких. В результате добавляется гибкость, которой нет в чистом водопаде, но и повышаются накладные расходы на планирование. При этом релиз на каждой итерации - это не просто новая версия продукта исправленная и улучшенная, а предыдущая версия вместе с еще одной частью конечного результата, которая не была разработана раньше. Т.е. изначально вся задача разбивается на множество мелких подзадач из которых и составляются релизы в итерационной парадигме. Это позволяет быстро произвести базовый продукт с ограниченной функциональностью и получить обратную связь о необходимом функционале намного раньше, чем это произошло бы в водопадной модели. Это позволяет, в том числе, изменить планы на дальнейшую разработку, вместо отката назад в случае, когда пользователю предоставляется продукт целиком.

Впрочем, у данной концепции есть и свои проблемы:

- Чем более мелкими получаются итерации при разбиении, тем больше возрастают организационные расходы. Эти расходы легко могут превысить возможные расходы при откате назад в более цельных моделях.

- Вместе с новой функциональностью в очередном релизе могут появиться интеграционные проблемы, говорящие об архитектурной ошибке при дизайне.

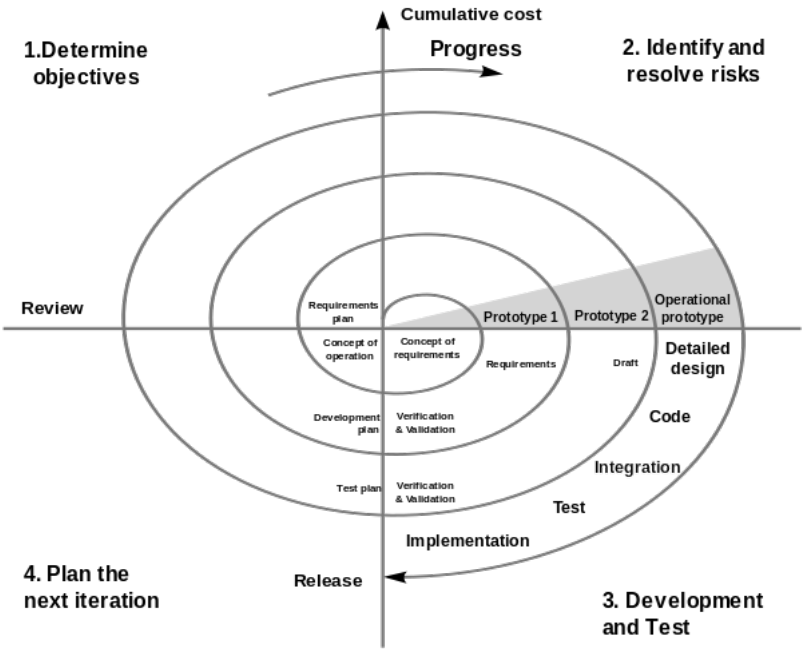
Для проекта распознавания данная концепция предоставляет некоторые механизмы, так как необходимая система в первом варианте достаточно сложна и многосоставна и, в результате, легко разбивается на части. Но эти части все равно остаются весьма большими и при их реализации необходимо проводить исследования, которых в данной методологии нет. К тому же, в определенный момент может быть принято решение о полном изменении модели на что-либо достаточно монолитное, в результате чего итерационный подход в его наивной реализации

полностью потеряет смысл. Далее будут рассмотрены более подходящие фреймворки, зачастую являющиеся развитием итерационной модели разработки.

Спиральная модель

Фактически не модель, а генератор моделей, так как позволяет, в зависимости от рисков проекта выбирать различные этапы из итеративной, водопадной или других моделей. При этом все модели, сгенерированные таким образом обладают некоторыми характерными чертами, инвариантами. Согласно [2] основными инвариантами спирально модели являются:

- Конкурентное определение основных артефактов (таких как требования, ключевые компоненты системы, ее дизайн, используемые алгоритмы и т.д.)
- Исполнение четырех базовых действий на каждом витке спирали:
  - Определение целей (выигрышных условий)
  - Определение и оценка альтернативных подходов к достижению целей
  - Определение и оценка рисков
  - Разработка, тестирование и планирование следующей итерации
- Риски определяют количество затрачиваемых усилий для любого действия в проекте
- Риски определяют степень детализованности любого действия в проекте
- Используются ключевые точки для определения прогресса и составления прогнозов
- Фокус на системе и ее жизненном цикле



Прототипирование

Развитием спиральной модели, равно как и, в некотором роде, итерационной можно считать модель разработки, основанную на создании прототипов конечного продукта. Каждый прототип сам по себе может реализовывать как полный функционал, так и всего лишь часть его.

Адаптивная разработка (Agile)

Парадигма разработки программного обеспечения, продвигающая адаптивное планирование, эволюционирующую разработку и мотивирующая на быструю реакцию при изменениях. Основные принципы согласно Agile Manifesto [9]:

- Быстрый и непрерывный релиз для удовольствия заказчика
- Изменения в требованиях приветствуются, даже на поздних стадиях разработки
- Рабочий продукт выходит регулярно (ближе к неделям чем к месяцам)

- Ежедневное тесное взаимодействие между бизнесом и разработчиками
- Проекты создаются мотивированными разработчиками, которым можно доверять
- Личное общение - лучший способ взаимодействия
- Работающий продукт - главный показатель успеха
- Устойчивый процесс разработки, проходящий в постоянном темпе
- Постоянное внимание к техническому совершенству и хорошему дизайну
- Простота - искусство максимизации работы, которую не нужно делать - одна из главных черт разработки
- Лучшая архитектура, дизайн и требования появляются в самоорганизующихся командах
- Регулярная рефлексия команды по поводу эффективности и последующие изменения с целью адаптирования.

Это замечательная парадигма, на основе которой создано множество методов разработки, позволяющая разрабатывать в условиях постоянного изменения требований, но в данном случае требования как раз более-менее постоянны, меняется подход к реализации. Следовательно разумнее взять одну из методик, предполагающих относительную неизменность требований и извлекающих из этого выгоду.

Итог

Для проекта в целом можно выбрать водопадную, итерационную или спиральную модель. Для текущего исследовательского этапа, на котором проверяются и сравниваются различные алгоритмы гораздо лучше подходит прототипирование.

Заключение

В рамках данной работы были получены следующие результаты:

- Проведена исследовательская работа и выбрана наиболее перспективная для дальнейшего изучения модель
- Проведены дополнительные исследования первой части выбранной модели и получено нестабильное решение
- Разработана и реализована система тестирования обученной системы
- Выбрана наиболее подходящая модель процесса разработки для проекта

Для дальнейшего развития полученных результатов необходимо провести дополнительные исследования всех частей выбранной модели и определение на основе данных исследований оптимальных параметров системы. После этого необходимо реализовать систему на языке более низкого уровня, предположительно C++ для максимального ускорения работы системы и достижения обработки видео в реальном времени.

Список литературы

[1] A Convolutional Neural Network Cascade for Face Detection / Haoxiang Li, Zhe Lin, Xiaohui Shen et al. // The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2015. — June.

[2] Boehm. Barry W. Spiral Development: Experience, Principles, and Refinements // Spiral Development Workshop. — 2000. — February.

[3] Dalal Navneet, Triggs Bill. Histograms of Oriented Gradients for Human Detection // In CVPR. — 2005. — P. 886–893.

[4] Evaluation of Traffic Sign Recognition Methods Trained on Synthetically Generated Data / Boris Moiseyev, Artem Konev, Alexander Chigorin, Anton Konushin // Advanced Concepts for Intelligent Vision Systems (Springer LNCS, Vol. 8192). — 2013. — P. 576–583. — URL: `http://link.springer.com/chapter/10.1007/978-3-319-02895-8_52`.

[5] Freund Yoav, Schapire Robert E. A Short Introduction to Boosting. — 1999.

[6] Gradient-Based Learning Applied to Document Recognition / Y. LeCun, L. Bottou, Y. Bengio, P. Haffner // Intelligent Signal Processing. — IEEE Press, 2001. — P. 306–351.

[7] ImageNet Large Scale Visual Recognition Challenge (ILSVRC). — URL: `http://image-net.org/challenges/LSVRC/` (online; accessed: 08.05.2017).

[8] LISA Traffic Sign Dataset. — URL: `http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html` (online; accessed: 08.05.2017).

[9] Beck Kent, Grenning James, Martin Robert C. et al. Manifesto for Agile Software Development. — 2001. — URL: `http://agilemanifesto.org/` (online; accessed: 08.05.2017).

[10] Mills H.D., Dyer M., Linger R.C. Cleanroom Software Engineering // IEEE Software. — 1987. — September. — Vol. 4. — P. 19 – 25.

[11] Sivic Josef, Zisserman Andrew. Video google: A text retrieval approach to object matching in videos // In ICCV. — 2003. — P. 1470–1478.

[12] Viola Paul, Jones Michael. Robust Real-time Object Detection // International Journal of Computer Vision. — 2001.

[13] Winston Dr., Royce W. Managing the Development of Large Software Systems // Proceedings of IEEE WESCON. — 1970. — August.

[14] Метрики в задачах машинного обучения. — URL: `https://habrahabr.ru/company/ods/blog/328372/` (дата обращения: 08.05.2017).

[15] Свёрточная нейронная сеть. — URL: `https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть` (дата обращения: 08.05.2017).